Finite-State Dirichlet Allocation: Learned Priors on Finite-State Models

Jia Cui and Jason Eisner
Department of Computer Science / Center for Language and Speech Processing
Johns Hopkins University
Baltimore, MD 21218 USA

Abstract

To model a collection of documents, suppose that each document was generated by a *different* hidden Markov model or probabilistic finite-state automaton (PFSA). Further suppose that all these PFSAs are *similar* because they are drawn from a single (but unknown) prior distribution over PFSAs. We wish to infer the prior, obtain smoothed estimates of the individual PFSAs, and reconstruct the hidden paths by which the unknown PFSAs generated their documents.

As an initial application, particularly hard for our model because of its sparse data, we derive an FSA topology from WordNet. For each verb, we construct the "document" of all nouns that have appeared as its object. Our method then learns a better estimate of $p(object \mid verb)$, as well as which paths in WordNet, and hence which senses of ambiguous objects, tend to be favored. Our method improves 14.6% over Witten-Bell smoothing on the conditional perplexity of objects given the verb, and 27.5% over random on detecting the most common senses of nouns in the SemCor corpus.

1 Introduction

1.1 Generative Models of Documents

Several recent machine learning papers have proposed fully generative models of document collections. This is useful if one wishes to reconstruct hidden portions of a document, or infer the underlying parameters of documents in order to cluster or classify them.

Such a model starts with hyperparameters α that describe the document collection. These are used to generate document-specific parameters θ for each document, which in turn generate the full text of that document. Both α and the θ's are unknown.

In the original Latent Dirichlet Allocation (LDA) model of [2], the document-specific parameters θ characterized only the mixture of topics in the document. The document text was then generated by a simple class-based unigram model.

In this paper, we go beyond unigrams to learn the *temporal* properties of each document. In our "Finite-State Dirichlet Allocation" (FSDA) model, the document-specific parameters describe the probabilities of labeled transitions in a finite-state generative process. Only the emission sequence Y is observed, but the hidden state sequence X can be reconstructed, which is useful.

There have been previous moves toward adding temporal structure to LDA. However, they did not learn *document-specific* temporal structure. We review this work in section 1.4.

1.2 Learning from Multiple Datasets

Another contribution of this paper is that we apply our generative model to something other than a document collection. Instead of learning how different documents generate their words, we attempt to learn how different verbs generate their direct objects by choosing paths through WordNet, which we regard as a finite-state process (following [1]).

Our point is that LDA and its relatives are applicable to multiple datasets that need not be documents. Our FSDA model has the further advantage that we can inject domain knowledge through the topology of the finite-state process, which in this case comes from WordNet.

Much machine learning uses a single dataset Y (not necessarily a document) with hidden labels X, and a family $p(X, Y \mid \theta)$ of probability distributions. We wish to estimate the particular parameter vector θ that was responsible for generating Y. Why? If we can guess θ (or better yet, a posterior probability distribution over θ), then we can use it to infer the hidden labels X. This is the basic setup in classic EM algorithms such as forward-backward, inside-outside, and Gaussian clustering.

A prior distribution can help smooth the estimate of θ. In other words, we may suppose that θ was itself sampled from, say, a Gaussian or a Dirichlet distribution, so that some θ vectors are *a priori* more likely than others. But how are we to decide *which* Gaussian or Dirichlet? In other words, how do we pick the so-called "hyperparameter vector" α that defines a *particular* prior in the chosen family?

A traditional engineering approach is to seek the α vector that yields the best performance, as evaluated on a small held-out dataset. However, this will not work well if α has many degrees of freedom.

There is, however, a special circumstance where we can infer α from training data. Suppose we have *multiple* datasets $Y^1, Y^2, \ldots Y^D$ (the "documents"), which are assumed to be respectively generated using some unknown $\theta^1, \theta^2, \ldots \theta^D$. Thus, there were *multiple* samples (the θ^d) from the same prior. It is therefore conceivable to infer the prior from these samples, or rather from our indirect evidence of them (the Y^d). The prior's mean will describe the "typical" θ^d, and its variance will describe how widely the different θ^d scatter about the mean. If the datasets appear to be underlyingly similar, then we will capture that fact by inferring a low-variance prior, which (for sparse Y^d) will smooth estimates of θ^d strongly back toward the mean.

Because the θ^d are not directly observed, this is a **two-level** inference problem: find α to maximize $\prod_{d=1}^{D} \int_{\theta^d} p(Y^d \mid \theta^d) \cdot p(\theta^d \mid \alpha)\, d\theta^d$. This approach is also known as "empirical Bayes." One could place a prior over α as well, but this paper assumes an uninformative (flat) prior throughout.

1.3 Two-Level Inference of Finite-State Models

Sequence data Y can often be modeled by a *finite-state* process. Estimating the process's parameters lets one predict future data, or reconstruct the hidden path X that underlies the observed sequence Y.

This paper addresses the two-level-inference version of the problem. We perform joint estimation on D finite-state processes, which are distinct but are presumed to be "similar" since they are all drawn from the same unknown prior. In other words, we fit a generative model of generative finite-state processes. Some possible applications:

- Each Y^d is a distinct document
- Each Y^d is a set of documents in a certain genre/topic
- Each Y^d records the sequential behavior of a certain user [5]
- Each Y^d is conditioned on a certain local context, e.g., it is a set of direct objects or syntactic subcategorization frames observed with a certain verb d

In each example, our method jointly estimates models of the D observed documents, genres, users, or verbs. At the same time, it estimates a prior that predicts what unobserved (or underobserved) documents, genres, users, or verbs will tend to look like.

Rather than use Hidden Markov Models, we adopt the similar framework of probabilistic finite-state automata (PFSAs). This lets us use ϵ transitions and stopping probabilities. ϵ transitions will be crucial. PFSAs are also simpler to notate, using only distributions over labeled transitions, rather than an HMM's separate distributions over transitions and emissions. Furthermore, they generalize naturally to finite-state transducers, which could be used to model unaligned *pairs* of sequences. A PFSA is an FSA where each state $s \in \{1, 2, \ldots S\}$ is equipped with a multinomial distribution $\theta_s \in \mathbb{R}^{K_s}$ over the K_s options for leaving that state: namely the outgoing arcs, plus the option of halting if the state is final. We write $\boldsymbol{\theta} = \langle \theta_1, \ldots \theta_S \rangle$ for the full set of parameters of the PFSA, using boldface to emphasize that it is a *sequence* of vectors, one per state.

Our model for **two-level** PFSA inference is shown in Figure 1. Each dataset Y^d is now a collection of strings that were generated by a particular PFSA defined by $\boldsymbol{\theta}^d$. We assume that all of these PFSAs have the *same* underlying S-state FSA, which is given in advance. As our prior, we assume that each θ_s^d is independently sampled from a Dirichlet distribution associated with state s: i.e., $\boldsymbol{\theta}^d \sim \text{Dirichlet}(\alpha_1) \times \cdots \times \text{Dirichlet}(\alpha_S)$, where each $\alpha_s \in \mathbb{R}^{K_s}$. We write boldface $\boldsymbol{\alpha} = \langle \alpha_1, \ldots \alpha_S \rangle$ for the full set of hyperparameters that define the prior.

Notice that we cannot freely simplify the underlying FSA with operations such as ϵ-elimination. That would alter the family of priors under consideration.

A Dirichlet is a smooth, unimodal distribution over multinomial distributions. A sample from Dirichlet(α_s) is a particular multinomial $\theta_s \in \mathbb{R}^{K_s}$ (so $\sum_i \theta_{si} = 1$ and $(\forall i)\theta_{si} \geq 0$). If $\alpha_s = \langle 5, 1, 3 \rangle$, then θ_s will tend to fall near $\langle 5/8, 1/8, 3/8 \rangle$; if $\alpha_s = \langle 50, 10, 30 \rangle$, then the Dirichlet is more sharply peaked, and θ_s will tend to fall even closer to that same mean. The full density is given by $P(\theta_s | \alpha_s) = \frac{1}{Z(\alpha_s)} \prod_{i=1}^{K_s} \theta_{si}^{\alpha_{si}-1}$, where $Z(\alpha_s)$ is a normalizing factor.

1.4 Prior Work

Two-level inference is difficult; our approximation methods for carrying it out draw on previous work.

In the discussion below, y_i, x_i, and s_i denote random variables, with only y_i being observed.

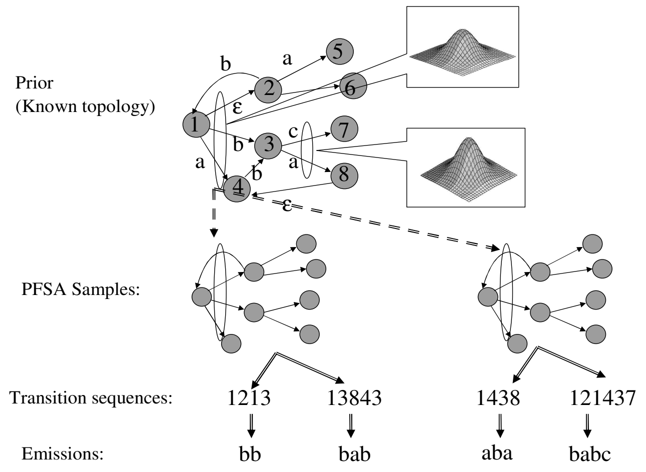

Figure 1: The FSDA model for generating documents. (1) A prior distribution (top) is given by a finite-state automaton equipped with a Dirichlet prior α_s at each state s. (2) The dth sample from this prior is a PFSA with a multinomial distribution θ_s^d at each state s. It will generate the dth document. (3) By taking random walks through this dth PFSA, we obtain a collection X^d of paths. (4) Deterministically reading the labels off the paths in X^d, we obtain a set of strings, which constitute the dth document Y^d. Only the Y^d are observed. Since the FSA has ϵ-transitions, even short observed strings can have long hidden paths.

[7] smooth a bigram language model $p(\vec{y})$ by inferring a single Dirichlet prior from which, for each word d in the vocabulary, the conditional distribution $p(y_i \mid y_{i-1} = d)$ was drawn. The mean of this prior can be regarded as a "typical" $p(y_i \mid y_{i-1} = \ldots)$ distribution toward which sparsely observed distributions are smoothed. By contrast, the traditional backed-off distribution $p(y_i)$ mainly reflects which words are used in *frequent* contexts (note that it equals $\sum_d p(y_i \mid y_{i-1} = d) \cdot p(y_{i-1} = d)$), and thus may not be as appropriate for smoothing distributions that are not as well-observed.

In our terminology, whenever $y_{i-1} = d$, MacKay and Peto let $y_i \sim \text{Multinomial}(\theta^d)$, where in turn $\theta^d \sim \text{Dirichlet}(\alpha)$. Evidence about θ^d (and hence indirectly about α) comes from the observed dataset Y^d, which is the sample of words y that were observed to follow d.

Latent Dirichlet Allocation [2] can be regarded as enriching the above with hidden variables. Now Multinomial(θ^d) does not generate y_i directly: it generates a hidden topic x_i from a set of K topics. The topic x_i then generates the observed word y_i. Concretely, $p(y_i \mid d) \overset{\text{def}}{=} \sum_{k=1}^{K} p(y_i \mid x_i = k) \cdot p(x_i = k \mid d)$. This reduces the number of parameters if K is small.

Applied to MacKay and Peto's problem, LDA thus restricts the next-word distribution $p(y_i \mid y_{i-1} = d)$ to be a mixture of K "topical" multinomials $p(y_i \mid x_i = k)$. These multinomials correspond to global senses or properties that different context words may draw on. For a given context word d, the K mixture coefficients $p(x_i = k \mid d)$ are given by a vector $\theta^d \in \mathbb{R}^K$ that was drawn from Dirichlet(α).

[5] (like us) reason that each of the D documents is not simply a bag of unigrams, but has higher-order sequential structure. Their Sequential Activity Profiling (SAP) is a "straightforward generalization of LDA." Each word y_i in document d is generated conditioned not only on the word's hidden topic x_i, but also on the previous $m-1$ words: $p(y_i \mid d) \overset{\text{def}}{=} \sum_{k=1}^{K} p(y_i \mid x_i = k, y_{i-1}, \ldots y_{i-m}) \cdot p(x_i = k \mid d)$. As before, $p(x_i = k \mid d)$ is defined by Multinomial(θ^d) where $\theta^d \sim$ Dirichlet(α).

[6] also model the temporal structure of documents. Unlike SAP, they use hidden states, which may capture a longer history than just the previous $m-1$ words. In their model, all documents are generated by the same HMM, with one twist. One of the HMM states is called "CONTENT WORD" and has a document-specific emission model based on LDA. The words emitted from this state are determined by a document-specific distribution over topics. This distribution is the *only* document-level parameter, just as in LDA.

Our FSDA method goes beyond all this previous work by learning *document-specific temporal parameters*. In FSDA, the transition probabilities and not just the emission probabilities can vary across documents. The amount that they tend to vary depends on the variance of the prior, $\boldsymbol{\alpha}$.

In a final piece of prior work, [4] shows how to estimate parameters of a single PFSA (of known topology) given a dataset and a prior. Our two-level inference procedure uses multiple datasets to estimate parameters of a *collection* of PFSAs *and* their common prior. On the other hand, Eisner considers a richer class of PFSA parameterizations. It would be interesting to extend our two-level procedure accordingly.

2 Parameter Estimation

2.1 Method 1: Viterbi-EM Approximation

Suppose that we did not have any hidden path (or topic) variables. That is, suppose we observed not only Y^d, a collection of sequences generated by the dth PFSA, but also X^d, the corresponding collection of paths through the PFSA.

Then [7], applied separately at each state s, would immediately solve our inference problem. For each d, we would collect the edges that were *observed* in X^d to leave s. Then MacKay and Peto's method would find estimates of α_s, as well as the posterior expectation of each θ^d_s.

This suggests a simple (but novel) method for inferring such estimates, using the Viterbi approximation to EM. Start with a guess of $\boldsymbol{\theta}$. Use it to reconstruct the most likely paths $X^1, \ldots X^D$ from $Y^1, \ldots Y^D$. Now apply MacKay & Peto's method to find the optimal $\boldsymbol{\alpha}$ and hence the posterior expectation of $\boldsymbol{\theta}$. Repeat until convergence.

The observed variables are the Y^d; the hidden variables are the X^d; and the parameters to optimize are the $\boldsymbol{\theta}^d$ and α. This method maximizes over $\boldsymbol{\theta}^d$.

While we have worked out the details of this Viterbi EM method, we did not implement it, feeling that sparse data called for integrating out $\boldsymbol{\theta}^d$ as promised in section 1.2. We now turn to an approximate method for doing that.

2.2 Method2: Variational Approximation

Despite the intractability of full EM, we now derive a variational approximation method. We follow the ideas detailed in section A.3 of [2], but generalize from LDA to two-level inference of PFSAs. In some ways, the generalization is simpler, because each Y^d is characterized by a single set of parameters $\boldsymbol{\theta}^d$, specifying the next state as well as output, rather than two sets as in LDA; Our exposition is also structured differently.

Our goal is to find $\boldsymbol{\alpha}$ that maximizes $\log \prod_d p(Y^d \mid \boldsymbol{\alpha})$. As we are now assuming X^d is unknown, and want to sum over all its possible values, this objective function equals

$$\sum_{d=1}^{D} \log \int_{\boldsymbol{\theta}^d} \sum_{X^d \in \pi(Y^d)} p(X^d \mid \boldsymbol{\theta}^d) \cdot p(\boldsymbol{\theta}^d \mid \boldsymbol{\alpha})\, d\boldsymbol{\theta}^d$$

where $\pi(Y^d)$ is the set of paths that can generate Y^d.

EM would require us to get the posterior distribution over the X^d and $\boldsymbol{\theta}^d$, which is intractable as discussed in the previous section. Intuitively, the problem is that each value of $\boldsymbol{\theta}^d$ yields a completely different distribution over paths X^d (notice that X^d may reuse the arc probability θ^d_{sk} many times). The variational EM idea is to instead maximize a lower bound that can be tractably decomposed (factored) whereas the true objective function cannot. The steps for choosing the best $\boldsymbol{\alpha}$ are as follows:

Variational E step: Given our current $\boldsymbol{\alpha}$, for *each* document Y^d, choose the tightest lower bound on $\log p(Y^d \mid \boldsymbol{\alpha}) \in \mathbb{R}$ that we can find. This bound will be specified by two sets of parameters, $\boldsymbol{\gamma}^d$ and $\boldsymbol{\phi}^d$, which fall in $\mathbb{R}^{K_1} \times \cdots \times \mathbb{R}^{K_s}$, just as $\boldsymbol{\alpha}$ and $\boldsymbol{\theta}^d$ do. (They will in fact serve as surrogates for $\boldsymbol{\alpha}$ and and $\boldsymbol{\theta}^d$—look at the symbols sideways—but they will take knowledge of the document Y^d into account.) These parameters are used to define a probability distribution $q(\boldsymbol{\theta}^d, X^d \mid Y^d, \boldsymbol{\gamma}^d, \boldsymbol{\phi}^d)$ from a preselected family of functions that is designed to be tractable yet yield good lower bounds. *Any* probability distribution q yields the following lower bound by treating its values as mixture coefficients in Jensen's inequality:

$$\begin{aligned}
\log p(Y^d|\boldsymbol{\alpha}) &= \log \int_{\boldsymbol{\theta}^d} \sum_{X^d \in \pi(Y^d)} p(\boldsymbol{\theta}^d, X^d, Y^d|\boldsymbol{\alpha}) \\
&= \log \int_{\boldsymbol{\theta}^d} \sum_{X^d \in \pi(Y^d)} q(\boldsymbol{\theta}^d, X^d|\ldots) \frac{p(\boldsymbol{\theta}^d, X^d, Y^d|\boldsymbol{\alpha})}{q(\boldsymbol{\theta}^d, X^d|\ldots)} \\
&\geq \int_{\boldsymbol{\theta}^d} \sum_{X^d \in \pi(Y^d)} q(\boldsymbol{\theta}^d, X^d|\ldots) \log \frac{p(\boldsymbol{\theta}^d, X^d, Y^d|\boldsymbol{\alpha})}{q(\boldsymbol{\theta}^d, X^d|\ldots)} \\
&= E_q[\log p(\boldsymbol{\theta}^d, X^d, Y^d|\boldsymbol{\alpha})] - E_q[\log q(\boldsymbol{\theta}^d, X^d|\ldots)]
\end{aligned}$$

Note that q must be defined to assign probability mass only to paths X^d that are consistent with the observed Y^d; otherwise the Jensen's mixture coefficients q would not sum to 1 (unless the summation were relaxed to range over all X^d, in which case the lower bound would reduce to an unhelpful $-\infty$).

At the E step, $\boldsymbol{\gamma}^d$ and $\boldsymbol{\phi}^d$ are optimized to maximize this lower bound, making it as tight as possible. The difference between the two sides of the inequality works out to be the Kullback-Leibler divergence $D(q(\boldsymbol{\theta}^d, X^d \mid Y^d, \boldsymbol{\gamma}^d, \boldsymbol{\phi}^d) \mid\mid p(\boldsymbol{\theta}^d, X^d | Y^d, \boldsymbol{\alpha}))$, so the resulting bound is tight to the extent that the resulting q is a good approximation of the *posterior* distribution p over parameters and paths (*given* the evidence and prior).

Variational M step: Given our current $\boldsymbol{\gamma}^1, \dots \boldsymbol{\gamma}^D, \boldsymbol{\phi}^1, \dots \boldsymbol{\phi}^D$, we choose $\boldsymbol{\alpha}$ to maximize the same lower bound considered above. More precisely, we maximize its sum over all $1 \leq d \leq D$. By factorizing the $p(\boldsymbol{\theta}^d, X^d, Y^d \mid \boldsymbol{\alpha})$ term as its definition $p(X^d, Y^d \mid \boldsymbol{\theta}^d) \cdot (\boldsymbol{\theta}^d \mid \boldsymbol{\alpha})$, and ignoring terms that do not depend on $\boldsymbol{\alpha}$, we see this may be done by maximizing

$$\sum_{d=1}^{D} E_q[\log p(\boldsymbol{\theta}^d \mid \boldsymbol{\alpha})]$$

Note that if q were a perfect approximation to posterior p at the E step, this would be precisely how an ordinary non-variational M step would choose $\boldsymbol{\alpha}$. We have simply made this maximization tractable by replacing p with q from an appropriate family, so that q can be expressed in a simple factored form, just as in the forward-backward algorithm.

The alternation of E and M steps can be regarded as an alternating maximization procedure on our formula that lower-bounds the true objective $p(Y^d \mid \boldsymbol{\alpha})$. Each step increases this lower bound. The E step does so by changing $\boldsymbol{\phi}^d, \boldsymbol{\gamma}^d$; the M step, by changing $\boldsymbol{\alpha}$.

We now describe our family q and the concrete instantiations of the E and M steps for this family. We define

$$q(\boldsymbol{\theta}^d, X^d \mid \boldsymbol{\gamma}^d, \boldsymbol{\phi}^d, Y^d) \stackrel{\text{def}}{=} q(\boldsymbol{\theta}^d \mid \boldsymbol{\gamma}^d) \cdot q(X^d \mid \boldsymbol{\phi}^d, Y^d)$$

where the first factor is defined as $\boldsymbol{\theta}^d \sim \text{Dirichlet}(\gamma_1^d) \times \cdots \times \text{Dirichlet}(\gamma_s^d)$, and X^d is a distributed as a random path from a PFSA with our usual structure but parameterized by $\boldsymbol{\phi}^d$ rather than $\boldsymbol{\theta}^d$.

However, by contrast with $\theta_s^d \in \mathbb{R}^{K_s}$, there is no requirement that each $\phi_s^d \in \mathbb{R}^{K_s}$ is a probability distribution over the arcs from state s. Such a requirement would turn the E step into a more difficult constrained optimization problem, and would rule out some q functions that might be good approximations to p. So the PFSA parameterized by $\boldsymbol{\phi}^d$ *is not actually a PFSA in the usual sense,* but rather a random field. It assigns each path X^d a positive weight (by multiplying ϕ values along the path) rather than a probability. These weights are then normalized to define q so that $\sum_{X^d \in \pi(Y^d)} q(X^d \mid \boldsymbol{\theta}^d, Y^d) = 1$. Formally,

$$q(X^d \mid \boldsymbol{\phi}, Y^d) = \prod_i (1/Z(\boldsymbol{\phi}^d, y_i^d)) \prod_{sk} \phi_{sk}^{c_{sk}(x_i^d)} \tag{1}$$

where $c_{sk}(X^d)$ is the number of times that path x_i^d traverses the kth arc from state s while generating string y_i^d.

The E-step iterative update of $\boldsymbol{\phi}^d$ and $\boldsymbol{\gamma}^d$ by maximizing the lower bound is:

$$\begin{pmatrix} \phi_{sk}^d \\ \gamma_{sk}^d \end{pmatrix} \leftarrow \begin{pmatrix} \exp(\Psi(\gamma_{sk}^d) - \Psi(\sum_{l=1}^{K_s} \gamma_{sl}^d)) \\ \alpha_{sk} + E_q[c_{sk}(X^d)] \end{pmatrix} \tag{2}$$

where

$$E_q[c_{sk}(X^d)] = \sum_{X^d} q(X^d \mid Y^d, \boldsymbol{\phi}^d) c_{sk}(X^d)$$

can be found by the forward-backward algorithm if the set of paths $\pi(y_i^d)$ is finite for each $y_i^d \in Y^d$. If the set of paths is infinite, because the FSA contains ϵ-cycles or y_i^d was not fully observed, the expectation can be found by solving a linear system of equations [4].

An arc which is never used to generate Y^d, thus having zero expectation, will still have nonzero ϕ, but this ϕ will not influence the value of q in (2), nor the update of $\boldsymbol{\gamma}^d$ in (2).[1]

In the M step, $\boldsymbol{\alpha}$ is updated depending only on $\boldsymbol{\gamma}^d$. The optimal value of $\boldsymbol{\alpha}$ can be obtained by satisfying the following equations using a Newton-Raphson algorithm:

$$D(\Psi(\sum_{k=1} \alpha_{sk}) - \Psi(\alpha_{sk})) + \sum_{d=1}^{D} (\Psi(\gamma_{sk}^d) - \Psi(\sum_{l=1} \gamma_{sl}^d)) = 0$$

The detailed derivation is similar to the one for LDA.

3 Inference

Once the generative model $p(y \mid \boldsymbol{\alpha})$ is trained, we wish to use it to infer the hidden path x for an observed string $y \in Y^d$. If we knew $\boldsymbol{\theta}^d$, we could simply run the Viterbi decoding algorithm to reconstruct $\arg\max p(x \mid y, \boldsymbol{\theta}^d)$. However, our only knowledge about $\boldsymbol{\theta}^d$ is its prior $\boldsymbol{\alpha}$ and the full document Y^d it generates. Therefore, we could either find the best $\boldsymbol{\theta}^d$ which is likely to be sampled from $\boldsymbol{\alpha}$ and generates Y^d or we rank all candidates paths according to their expected probability given the distribution of $\boldsymbol{\theta}^d$.

To get the best $\boldsymbol{\theta}^d$, we maixmize $p(Y^d \mid \boldsymbol{\theta}^d) p(\boldsymbol{\theta}^d \mid \boldsymbol{\alpha})$. This optimization requires another EM run, which is easy to derive. Note the first part is just a normal PFSA and the E-step is a normal E-step depending only on $\boldsymbol{\theta}^d$. In M-step, $\boldsymbol{\theta}^d$ is optimized based on both expectations of hidden paths and the prior: $\boldsymbol{\theta}_{sk} \propto E_{\boldsymbol{\theta}'}[c_{sk}(X^d)] + \alpha_{sk} - 1$ where $\boldsymbol{\theta}'$ denotes parameters in the previous iteration.

When inferring the optimal $\boldsymbol{\theta}$, we shift away from our original goal of smoothing $p(Y^d|\boldsymbol{\theta}^d)$ by $\int_{\boldsymbol{\theta}^d} p(Y^d|\boldsymbol{\theta}^d) p(\boldsymbol{\theta}^d|\boldsymbol{\alpha}) d\boldsymbol{\theta}^d$. To smooth over all possible $\boldsymbol{\theta}^d$ as well as taking Y^d into account, we select

[1]We could alternatively have presented ϕ as specifying, for each i, an individual weight for each arc in the "trellis" of all FSA paths that generate y_i^d. The trellis is obtained by intersecting the FSA with the string y_i^d. Multiple trellis arcs derived from the same FSA arc would then have separate parameters, but the parameters would turn out to have equal values at the optimum. If a particular FSA arc goes unused by Y^d and yields no trellis arcs, then no parameters would be associated with it. This presentation would be similar to the LDA presentation in [2], where longer documents give rise to more ϕ parameters, but multiple copies of the same word in a given document end up having the same parameter value.

x to maximize $p(x, y \mid Y^d, \boldsymbol{\alpha})$ for given y.

$$\begin{aligned}
p(x, y \mid Y^d, \boldsymbol{\alpha}^d) &= \int_{\boldsymbol{\theta}^d} p(x, y \mid \boldsymbol{\theta}) p(\boldsymbol{\theta}^d \mid Y^d, \boldsymbol{\alpha}^d) d\boldsymbol{\theta}^d \\
&\approx \int_{\boldsymbol{\theta}^d} p(x, y \mid \boldsymbol{\theta}^d) q(\boldsymbol{\theta}^d \mid \boldsymbol{\gamma}^d) d\boldsymbol{\theta}^d \\
&= \int_{\boldsymbol{\theta}^d} \prod_{sk} (\theta^d_{sk})^{c_{sk}(x)} \prod_s \frac{1}{Z(\gamma^d_s)} \prod_k (\theta^d_{sk})^{\gamma_{sk}-1} d\boldsymbol{\theta}^d \\
&= \prod_s \frac{Z(\gamma^d_s + c_s(x))}{Z(\gamma^d_s)} \int_{\theta^d_s} \frac{\prod_k (\theta^d_{sk})^{c_s k(x) + \gamma_{sk} - 1}}{Z(\gamma^d_s + c_s(x))} d\theta^d_s \\
&= \prod_s \frac{Z(\gamma_s + c_s(x))}{Z(\gamma_s)} \qquad (3)
\end{aligned}$$

where $c_s(x)$ is a vector with the kth element being $c_{sk}(x)$. Usually, $c_s(x)$ at different states have complex dependent relationship with each other. But when there is no loop in the structure, $c_s(x)$ are independent. They are vectors with only one element being 1 and the others being zero. In this case, the optimization can be solved by dynamic programming in linear time. At any time when we reach a state s, the next step depends only on arcs started from s. The weight from s to k' is

$$\frac{Z(\gamma_s + c_s(x_{k'}))}{Z(\gamma_s)} = \frac{\prod_{k \neq k'} \Gamma(\gamma_{sk}) \Gamma(\gamma_{sk'} + 1)}{\Gamma((\sum_k \gamma_{sk}) + 1)} \frac{\Gamma(\sum_k \gamma_{sk})}{\prod_k \Gamma(\gamma_{sk'})} = \frac{\Gamma(\gamma_{sk'} + 1) \Gamma(\sum_k \gamma_{sk})}{\Gamma(\gamma_{sk}) \Gamma(1 + \sum_k \gamma_{sk})}$$

This criterion could also be applied in cases when there are loops in the PFSA but we have limited number of finite candidate x to choose from.

Both above inference methods are originated from exploring $\boldsymbol{\alpha}$. But just as $\boldsymbol{\gamma}^d$ can be used for approximation, $\boldsymbol{\phi}^d$ can be used in some cases to simplify the tagging process too. If we rewrite $p(x \mid y, Y^d, \boldsymbol{\alpha}^d)$ as $\int_{\boldsymbol{\theta}^d} p(x, \boldsymbol{\theta}^d \mid y, Y^d, \boldsymbol{\alpha}) d\boldsymbol{\theta}^d$ and take approximation $p(x, \boldsymbol{\theta}^d \mid y, Y^d, \boldsymbol{\alpha}) \approx q(\boldsymbol{\theta}^d \mid \boldsymbol{\gamma}^d) q(x \mid y, Y^d, \boldsymbol{\phi}^d)$, we get $p(x \mid y, Y^d, \boldsymbol{\alpha}^d) \approx q(x \mid y, Y^d, \boldsymbol{\phi}^d)$. If $y \in Y^d$, we could use $\boldsymbol{\phi}^d$ obtained in the training process for tagging directly. But according to the definition of q, this method works only for y which belongs to Y^d, because arcs which are not used by Y^d have unconstrained values.

4 Applications

Given an FSDA model, we could do the following things:

1. Infer the unigram distribution $p(Y \mid \boldsymbol{\alpha})$.
2. Infer the bigram distribution $p(Y \mid d, \boldsymbol{\alpha})$.
3. Infer the path distribution given the observation Y: $p(X \mid Y, \boldsymbol{\alpha})$.
4. Infer the path distribution given the observation Y and the document type d: $p(X \mid d, Y, \boldsymbol{\alpha})$.

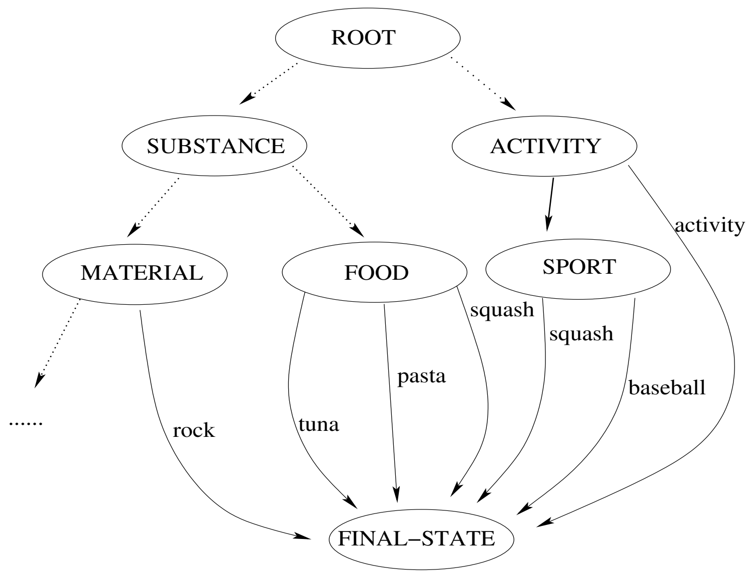

Figure 2: A finite-state machine corresponding to a tiny, simplified portion of WordNet. The unlabeled arcs are epsilon transitions. (The full machine is acyclic (though that is not required by our method) but not a tree: for example, PERSON can be reached via either LIFE-FORM or CAUSAL-AGENT [1], so "man" is a likely object of both "kill" and "help.")

Our experiments thus contain two parts: inferring the distribution of observed data as in 1. and 2., and inferring the hidden paths that underlie the observed data as in 3. and 4. In our task domain (described in the next section), the former experiments build a model of $p(directobject \mid verb)$, which we evaluate by bigram perplexity; such models are useful in statistical parsing. The latter experiments try to infer hidden paths through WordNet that correspond to hidden word senses; in short, they attempt unsupervised word sense disambiguation.

4.1 WordNet as a Finite-State Model

A useful resource in modeling $p(object \mid verb)$ and $p(sense \mid object)$ is WordNet [9], a hierarchical taxonomy of 94503 English nouns.

Figure 2 illustrates how we regard the directed acylic graph of WordNet as a sparse FSA, an idea due to [1]. Each root-to-leaf path generates an English noun.

We presume that an instance of "eat" selects its direct object by taking a random walk in a *probabilistic* FSA (PFSA) that has the structure of Figure 2. Starting at ROOT, it first decides stochastically whether to eat a SUBSTANCE or something else. Once it has chosen to eat a substance, further decisions specialize its choice further until it has decided to eat "squash." Verbs other than "eat" will have to make the same choices, but will use different probabilities reflecting their distinctive selectional preferences. Our prior treats all nodes as independent, since we suspect even verbs with strong preferences at some nodes will have "typical" distributions at other nodes: for example, "buy" is less likely to choose FOOD than "eat" is, but once we have chosen FOOD, the distribution of bought foods resembles the distribution of eaten foods.

If we can estimate the parameters $\boldsymbol{\theta}^{\text{eat}}$ of "eat"'s PFSA, we will obtain a probability distribution over direct objects. If our training sample Y^{eat} contains (eat,pizza), (eat,pasta), and (eat,squash), then we might conclude that "eat" has a preference for direct objects that fall under the SUBSTANCE node of WordNet and particularly its descendant FOOD. This lets us predict that $p(\text{tuna} \mid \text{eat}) > p(\text{rock} \mid \text{eat}) > p(\text{baseball} \mid \text{eat})$, which is useful for parsing and language modeling.

Even more interestingly, we will obtain a probability distribution over the hidden paths X^{eat} in the PFSA. Notice that Figure 2 includes two paths that generate "squash." We associate each path with a different *sense* of that word. If we observe (eat, squash) in test data, we may guess the intended sense by observing that the "eat" PFSA assigns a higher overall probability to the path through FOOD. By contrast, if we observe (play, squash), the parameters of the "play" PFSA tell us that the direct object was more likely generated by the other path, through SPORT.

Applying our methods will estimate the hyperparameters $\boldsymbol{\alpha}$. As for $\boldsymbol{\theta}^d$, our variational EM approximation finds $\boldsymbol{\gamma}^d$ such that $q(\boldsymbol{\theta}^d \mid \boldsymbol{\gamma}^d)$ approximates $\boldsymbol{\theta}^d$'s full posterior distribution (given the prior $\boldsymbol{\alpha}$ and the evidence Y^d). Our variational approximation also estimates a posterior distribution (1) over the hidden path X^d in *training* data.

Although each direct object $y \in Y^d$ is generated independently of the others, which suggests unigrams, it is actually generated by some path x that is labeled with $\epsilon\epsilon\ldots\epsilon y$. The temporal structure that we are modeling is the structure of these hidden paths, which crucially represent the hidden senses of words that we are trying to recover.

4.2 Predicting objects given verbs

In this section, we build a model of $p(directobject \mid verb)$, which we sometimes abbreviate as $p(n|v)$. Because most of our training verbs are only observed with a few of the many nouns in WordNet, we prune the enormous WordNet structure on a per-verb basis. The appendix describes how we reconcile this shortcut with the FSDA approach, which assumes that all verbs use an FSA of the same topology.

Data preparation We have the (verb,object) pairs extracted from BNC data by [1]. We use only the 486 verbs with count over 300, which give a total of 806K their (verb,object) tokens. We split these tokens randomly, taking 90% for training the language model and 10% for test.

The vocabulary contains all verbs and objects in the training data. It is the same for our baseline and the FSDA modeling.

Language model As noted in section 4.2, we prune WordNet differently for each verb when modeling $p(n \mid v)$. The goal is to keep as small as possible the structure for each verb FSA and meanwhile ensure all nodes are either not used or in the same shape across verbs. The background ($\boldsymbol{\alpha}$) structure contains all nodes and edges used by the objects in the training data. FSA for verb d is the substructure used by objects following d in the training data. In order to assign non-zero probability of words not covered in the FSA, an UNK node is added to that FSA. The probability of the UNK node is the total probability mass to be assigned to all nouns which are not covered by the FSA. This probability mass is used to backoff to the background unigram distribution.

Corresponding to the structure we described, we defined 4 sets of objects: V is the vocabulary, used by SRI and LDA V_α contains all objects seen in the training data. $V_\gamma(v)$ contains all objects which follow v in training data, and their sisters which are also in V_α. $T(v)$ is contains all objects following v in training data.

For easy annotation, we define four un-intersected sets including $T(v)$: $U_\gamma(v) = V_\gamma(v) - T(v)$, $U_\alpha(v) = V_\alpha - V_\gamma(v)$ and $U = V - V_\alpha$.

The FSDA language model will assign probability to words belonging to the first two sets with PFSA of verbs and assign probabilities to words in the remaining sets with the background model.

The FSDA language model is described as follows:

$$p(n|v) = \begin{cases} P(n|v) & n \in T(v) \\ P(n|v) & n \in U_\gamma(v) \\ P(\text{“}UNK''|v)q(n|v) & otherwise \end{cases}$$

$$q(n|v) = \begin{cases} (1 - \delta(v))\frac{Q(n)}{\sum_{w \in U_\alpha(v)} Q(w)} & n \in U_\alpha(v) \\ \delta(v)1/|U| & n \in U \end{cases}$$

where $P(n|v)$ is the probability of generating n from v's FSDA equipped with a real distribution, $Q(n|v)$ is the probability of generating n from the α structure equipped with a real distribution. $\delta(v)$ is an discount coefficiency. $\delta(v) = \frac{|U_\alpha(v)|}{|U_\alpha(v)| + \sum_{w \in U_\alpha(v)} count(w)}$. $\delta(v)$ never changes during the training process.

Perplexity results For each verb d, we would like to obtain its distribution over paths, and hence over direct objects, by integrating over the posterior distribution of $theta^d$. This posterior distribution is conditioned on both the trained α and all the training data, so as before, the integral is intractable. We simply rely again on the same approximation to the posterior that we used during training: thus, we construct a single FSA for verb d by taking the transition distribution from each state s to be the mean of Dirichlet($\boldsymbol{\gamma}_s^d$).

This model exploits the structure of WordNet. We compare it with standard baseline models estimated from the same data that do not use WordNet: $p(n \mid v)$ is smoothed back to $p(n)$ using Kneser-Ney or Witten-Bell smoothing. These models are constructed using the SRILM toolkit. As another baseline, we estimate a Latent Dirichlet Allocation model, again regarding each verb's set of direct objects as a document; again this does not use WordNet, but like WordNet it recognizes that each noun may have multiple senses and that multiple nouns may share a sense. LDA obtained the best performance with $k = 80$ topics.

Our FSDA method improves perplexity by 14.7% over the best of these comparisons (Witten-Bell). All results are presented in Table 1.

In order to better understanding why FSDA can beat Witten-Bell smoothing and LDA, we present perplexities of different type of predictions in Table 2. $S1$ includes (verb,object) pairs which are observed in the training data. $S2$ includes (verb,object) pairs which are not observed in the training data but the object can be predicted from the FSA of the verb. $S3$ includes

Model	Perplexity or Lowerbound
Kneser-Ney (SRILM)	1064
Witten-Bell(SRILM)	1051
LDA bigram-lowerbound	1183
LDA bigram	1171
FSDA bigram	896

Table 1: Perplexity of $p(object \mid verb)$ from different model

predictions by the background model and the objects are seen in the training data. $S4$ is the set of OOV predictions.

category	% of test	Witten-Bell	FSDA
$S1$	79.50	379	389
$S2$	10.49	54983	15648
$S3$	9.54	55067	33773
$S4$	0.46	64873	35399

Table 2: Perplexity for different prediction

Table 2 shows that FSDA has much better predictions on the unseen (verb, noun) pairs, especially on data set $S2$ where the nouns are not seen to follow the verb in the training data, but have similar senses to the seen objects. This improvement should be attributed to the WordNet structure used in FSDA modeling.

4.3 Word Sense Disambiguation

In this subsection, we describe how to train our FSDA for unsupervised word sense disambiguation. The structure of PFSA and the background models are the same as described in the language modeling part (see again the Appendix). However, we are more careful about initialization because there are many local optima in our training process, and we do not wish our initialization to favor one path (sense) over another for a given noun.

There are several ways to initialize the parameters of variational training.

1. initialize all alpha value as 1.

   This initialization favors short paths.

2. distribute the counts of all objects among the nodes properly so that if counts of edges are normalized at each node as transition probabilities, all senses of an object are equally likely. We call this method of distributing counts among edges as a "fair" count allocation.

   If we use these counts directly as $\boldsymbol{\alpha}$, $\boldsymbol{\alpha}$ would have too low variance than expected. If we normalize counts at each node as $\boldsymbol{\alpha}$, it is unclear what's the best way to do it.

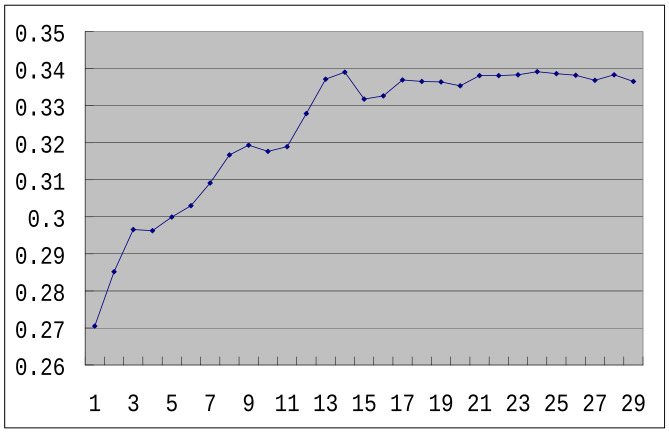

Figure 3: Accuracy at each iteration in variational training

3. For each verb, do a "fair" count allocation with smoothed counts of its objects. Then use these counts as $\boldsymbol{\gamma}^d$ and use these $\boldsymbol{\gamma}^d$ to estimate an $\boldsymbol{\alpha}$ as the initialized $\boldsymbol{\alpha}$.

   This initialization supports that $\boldsymbol{\alpha}$ at nodes which have similar distributions in different verb FSA will have low variance and vice versa.

A natual way of initializing $\boldsymbol{\phi}^d$ is to first apply a "fair" count allocation with smoothed counts of objects of this verb and then normalized these counts at each node. This ensures that the weight of all senses of an object are equal.

Word sense disambiguation on SemCor corpus This task uses a much smaller dataset—the SemCor corpus, where the true WordNet senses of the nouns are known (for evaluation purposes). We use the Minipar parser to extract 10k distinguished (verb,noun) pairs, including 3.5k noun types. All these data are used for training, but only ambiguous nouns are used in testing. There are about 7.6k distinguished test (verb,noun) pairs with 2.2k ambiguous noun types.

The first version of our task is to disambiguate nouns while using no context whatsoever at test time. This is the task that [8] refer to as First Sense: discovering without supervision which of the senses in WordNet are predominant in a given corpus. The random baseline, uniformly selecting among the possible senses, is 26.5%. If we use the mean of the Dirichlet distribution with the initialized $\boldsymbol{\alpha}$ as transition distribution, the result is 27.1%. After our variational FSDA training, it converges to 33.8% as showed in Figure 3. This shows that the model is extracting useful hidden paths. (However, the performance is far less than the 60% obtained by [8], whose clustering-based method used much richer context in *training* than just the single governing verb that we use.)

The second version of our task is to disambiguate a noun token whose governing verb d is known. This can be done by using the trained $\boldsymbol{\gamma}^d$ to select paths favored by nouns. Unfortunately, our result actually gets worse (29.0%) when we consider this additional context! Data analysis shows that 831 out of 1701 verbs in the training data have only one object type, making this an especially challenging task for the FSDA model. We believe that the sparsity of our observations causes the FSDA to overestimate the variance (from verb to verb) of the transition distributions from a rarely observed state, and therefore to smooth very little across verbs. In short, the FSDA model did not see enough data to discover how to smooth the data that it did see. A solution would be to impose an external prior on α_s, to reflect our prior belief that verbs that select for VEGETABLE won't differ much as to which particular vegetables they like. Indeed, we found that performance was improved by an ad hoc manipulation of $\boldsymbol{\alpha}$ during training to lower its variance. In the final version of this paper, we will evaluate the use of a real prior for all tasks.

WSD on Resnik test data [2]

[10] built a small training dataset and hand-labeled test dataset that were subsequently used by others [1, 3] for evaluating WordNet-based methods of learning selectional preferences from (verb,object) pairs.

We caution that this training dataset is curiously small for an unsupervised method (< 400 tokens). Also, Resnik deliberately restricted to the verbs with the strongest selectional preferences, so methods that can manage to learn from a wider range of verbs are not rewarded.

[1] separately learned PFSA parameters $\boldsymbol{\theta}^d$ for each verb d, smoothing each θ_s^d toward a uniform distribution. This is equivalent to fixing each α_s^d at $(1,1,1,\ldots,1)$. Of course, that is not necessarily the appropriate prior. Our FSDA method tries to *learn* the prior α_s^d, *generalizing across verbs* to determine typical probabilities for the different children of s (the mean of the Dirichlet) and how strongly typical they are (the variance of the Dirichlet). This is difficult because the paths are hidden, the EM problem requires a variational approximation, and we suffer from overfitting, but it does allow us to outperform Abney and Light (Table 3).

[3] built a Bayesian network whose topology, like our PFSA's topology, mirrored WordNet. Like Abney and Light, they assumed simple fixed parameters for their model and treated each verb entirely separately. They tried to explain each verb's pattern of observed nouns by inferring a set of WordNet nodes that were explicitly "selected" by that verb. (Strikingly, they did not consider the strength of selection.) Our method did not do quite as well as theirs.

Because the above methods attach fixed parameters to every node, they are sensitive to the topology of WordNet. Short paths and nodes with few children end up introducing a bias in favor of some senses over others. Both papers above were able to improve their performance by correcting for such imbalances in an ad hoc way (Table 3). We note for the sake of principle that our method does not suffer from these imbalances in the first place: it learns all its parameters and hence is "self-balancing."

[1] also tried a more realistic unsupervised training set extracted from the British National Corpus. They kindly provided us with 1 million (verb, object) pairs. Retesting on Resnik's hand-labeled test data, we again get higher performance (rightmost column of Table 3), but the comparison is not fair because they were only able to train on a subset of these pairs.[3]

[2]Initialization and smoothing are different in the following two paragraphs.

[3]We were unable to compare on this subset because it is no longer known.

Model	Resnik	BNC
Random	28.5%	28.5%
Resnik	44.3	-
Abney&Light (w/o balancing)	35.6	36.5
Abney&Light (w/ balancing)	42.3	54.2
FSDA	44.8	55.2
Ciaramita&Johnson (w/o balancing)	45.6	-
FSDA, oracle stopping criterion	48.3	58.6
Ciaramita&Johnson (w/ balancing)	51.4	-
First Sense of noun type (supervised)	82.7	82.7

Table 3: Word-sense disambiguation accuracy on the Resnik test set, with two different training datasets.

Dominant senses We also evaluated whether our learned $\boldsymbol{\alpha}$, which describes typical paths through WordNet, successfully captured the *dominant* senses of the noun *types*. This task was recently addressed by [8]. Their method is entirely different in character from ours but makes use of the same resources (parsed BNC data and WordNet). They predict that x is the dominant sense for y if many words that tend to appear in the same syntactic contexts as y admit senses close to x in WordNet.

Like McCarthy et al., we trained our model unsupervised on syntactic dependencies extracted from the full British National Corpus. However, we had less training data because we only trained on the (verb,object) pairs, whereas they also trained on other dependencies such as (verb,subject) and (noun+preposition,object). We could in principle do this as well, training and testing on noun tokens from a greater variety of syntactic contexts.

To predict the dominant sense of a noun n, we asked which sense would be most likely under our model if n appeared with a *novel* verb. This depends on $\boldsymbol{\alpha}$.

Like McCarthy et al., we used our predicted dominant senses to tag all noun tokens in the SENSEVAL-2 English all-words corpus. The disappointing results are shown in Table 4. Recall that our initial model *is* a random classifier, because of the way we initialize $\boldsymbol{\alpha}$. As the stopping criterion stopped learning quickly, we did not learn to be much different from the initial model.

Method	Precision	Recall
Random	47.7%	47.7%
FSDA dominant sense	50	50
McCarthy dominant sense	64	63
WordNet first sense (supervised)	69	68
True dom. sense if any (oracle)	92	72

Table 4: WSD on Senseval-2 English all-words corpus, when tagging each token with its type's predicted dominant sense. The first two rows use the actual dominant sense from the test data and from SemCor, respectively. The next two rows use unsupervised learning.

5 Conclusions

We have presented a generative model of strings (e.g., sentences or paths through WordNet), shown in Figure 1. The novel feature is that different groups of strings (e.g., documents, or objects of different verbs) exhibit different internal temporal structure because each group is generated by a slightly different finite-state process. We have also presented algorithms for jointly estimating the parameters of all these processes.

Our model improves in a natural way on the unigram models of Latent Dirichlet Allocation, by adding document-specific temporal structure.

By casting our model in terms of probabilistic finite-state automata rather than HMMs, we allowed finite-state topologies with arbitrary ϵ-transitions and even ϵ-cycles. This means that the hidden path may be arbitrarily longer than the observed sequence. The importance of such topologies was illustrated by our experimental application to word sense disambiguation, in which *most* transitions were silent ϵ-transitions.

Our method also applies naturally to incomplete data, where the observation Y^d is not a fully observed document (string) but may be any regular language in which the document was observed to fall. As before, Y^d simply serves to determine the set of possible paths X^d.

Our experimental problem, learning selectional preferences via WordNet, was inspired by a desire to try a more principled version of Abney and Light's [1] "WordNet as HMM" technique. In order to have a better evaluation, we tested on the whole SemCor corpus and compared with the random baseline. We were able to improve on the first sense detection. But when verb information was added, the word sense disambiguation performance was worse than using the automatically extracted first sense. Preliminary results showed that imposing a prior on α might help by preventing overfitting.

Another motivation of the FSDA model was [7]. They used Dirichlet prior to smooth the bigram language model. Our FSDA model went further by adding WordNet structure as a latent layer. The experimental results showed that in the case of predicting direct objects given verbs, this WordNet structure improved the performance considerably.

We believe that the FSDA model itself, and the algorithms we have derived for it, are interesting and useful contributions to the NLP/ML literature. In future work, we plan to try them on more manageable tasks with fewer free parameters and hidden variables (see section 1.3). We also point out a natural generalization. Imagine fitting the parameters $\boldsymbol{\theta}$ of a large PFSA, under the assumption that certain sets of multinomials $\{\theta_s : s \in \overline{s}\}$ were all drawn from the same unknown prior $\alpha_{\overline{s}}$, where $\overline{s}$ is a given equivalence class of states and K_s is the same for all $s \in \overline{s}$. We leave it to the reader to see why this generalizes our framework, and why the more general framework would allow (among other things) training a model of genres on strings whose genres (Y^d) are not known in advance.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 0347822. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] Steven Abney and Marc Light. Hiding a semantic hierarchy in a Markov model. In *Proceedings of the ACL'99 Workshop on Unsupervised Learning in Natural Language Processing*, pages 1–8, 1999.

[2] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, January 2003.

[3] Massimiliano Ciaramita and Mark Johnson. Explaining away ambiguity: Learning verb selectional preference with Bayesian networks. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING)*, pages 187–193. Association for Computational Linguistics, 2000.

[4] Jason Eisner. Parameter estimation for probabilistic finite-state transducers. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Philadelphia, July 2002.

[5] Mark Girolami and ATa Kabán. Sequential activity profiling: Latent Dirichlet allocation of Markov chains. *Data Mining and Knowledge Discovery*, 2005.

[6] T. Griffiths, M. Steyvers, D. Blei, and J. Tenenbaum. Integrating topics and syntax. In *Advances in Neural Information Processing Systems (NIPS) 17*, 2004.

[7] David MacKay and Linda Peto. A hierarchical Dirichlet language model. *Journal of Natural Language Engineering,*, 1(3):1–19, 1995.

[8] Diana McCarthy, Rob Koeling, Julie Weeds, and John Carroll. Finding predominant senses in untagged text. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 280–287, Barcelona, Spain, 2004.

[9] George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J. Miller. Introduction to WordNet: An on-line lexical database. *International Journal of Lexicography*, 3(4):235–244, 1990.

[10] Philip Resnik. Selectional preference and sense disambiguation. In *Proceedings of the ANLP-97 Workshop: Tagging Text with Lexical Semantics: Why, What, and How?*, Washington, D.C., 1997.